IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Applicant:	§	
	§	
MICHAEL KAGAN ET AL.	§	Confirmation No. 9182
	§	
Serial No.: 10/000,456	§	
	§	
Filed: December 4, 2001	§	Group Art Unit: 2451
	§	
For: NETWORK INTERFACE	§	Attorney
ADAPTOR WITH DATA	§	Docket: 3091/24
SEND RESOURCES	§	
	§	
Examiner: Kamal B. Divecha	§	

Mail Stop Appeal Brief-Patents
Commissioner of Patents
Alexandria VA 22313-1450
ATTENTION: Board of Patent Appeals and Interferences

APPELLANT'S BRIEF

Dear Sir:

This is in furtherance of the Notice of Appeal filed in this case on October 27, 2009.

The fees required under § 1.17(f) and any required petition for extension of time for filing this brief and fees therefor are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief contains these items under the following headings and in the order set forth below:

I.   REAL PARTY IN INTEREST

II.   RELATED APPEALS AND INTERFERENCES

III.   STATUS OF CLAIMS

IV.   STATUS OF AMENDMENTS

I. REAL PARTY IN INTEREST

The real party in interest in this case is:

Mellanox Technologies Ltd.

P. O. Box 586

20692 Yokneam

ISRAEL

II.   RELATED APPEALS AND INTERFERENCES

NONE

III.   STATUS OF CLAIMS

A.   TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-66

B.   STATUS OF ALL THE CLAIMS

1.   Claims cancelled: 2, 3, 10, 13, 15, 32, 33, 40, 43, 45, 60-63

2.   Claims withdrawn from consideration but not cancelled: 20-30, 50-59

3.   Claims pending: 1, 4-9, 11, 12, 14, 16-19, 31, 34-39, 41, 42, 44, 46-49, 64-66

4.   Claims allowed: NONE

5.   Claims rejected: 1, 4-9, 11, 12, 14, 16-19, 31, 34-39, 41, 42, 44, 46-49, 64-66

C.   CLAIMS ON APPEAL

The claims on appeal are: 1, 4-9, 11, 12, 14, 16-19, 31, 34-39, 41, 42, 44, 46-49, 64-66

IV.   STATUS OF AMENDMENTS

An amendment to present the rejected claims in better form for consideration on appeal was filed on November 19, 2009.

V.   SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 is directed towards a network interface adapter (Figure 1, host channel adapter 22) that comprises a host interface (Figure 2, translation protection table 58 and doorbells 62), an outgoing packet generator (Figure 2, execution engine 60 and send data engine 66), a network output port (Figure 2, output port 68), a network input port (Figure 2, input port 50) and an incoming packet processor (Figure 2, transport check unit 52 and receive data engine 56). The host interface is for coupling to a host processor (Figures 1 and 2, CPU 24). The outgoing packet generator is adapted to generate an outgoing request packet for delivery to a remote responder (Figure 1, remote host 30 or I/O device 34) responsive to a request submitted by the host processor via the host interface (page 19 lines 19-24, page 20 lines 12-15, page 21 lines 8-10, page 22 line 30 through page 23 line 9). The network output port is coupled to receive the request packet from the outgoing packet generator so as to transmit the outgoing request packet over a network (Figures 1 and 2, IB fabric 26) to the remote responder (page 20 lines 15-19, page 21 line 29 through page 22 line 1). The network input port is for coupling to the network so as to receive an incoming response packet from the remote responder, in response to the outgoing request packet sent thereto, and to receive an incoming request packet sent by a remote requester (Figure 1, remote host 30) (page 18 lines 27-29, page 21 lines 20-23). The incoming packet processor is coupled to the network input port so as to receive and process both the incoming response packet and the incoming request packet (page 18 line 29 through page 19 line 7). The incoming packet processor also is coupled to cause the outgoing packet generator, responsive to the incoming request packet, to generate, in addition to the outgoing request packet, an outgoing response packet for transmission via the network output port to the remote requester (page 19

7

line 27 through page 20 line 7). The outgoing request packet comprises an outgoing write request packet containing write data taken from a system memory (Figures 1 and 2, system memory 38) accessible via the host interface (page 19 lines 12-19, page 23 lines 3-7). The outgoing response packet comprises an outgoing read response packet containing read data taken from the system memory in response to the incoming request packet (page 20 lines 12-15, page 21 lines 15-16). The incoming request packet comprises an incoming read request packet specifying data to be read from a system memory accessible via the host interface (page 3 lines 1-3, page 25 lines 6-10). The incoming packet processor is adapted to write a response descriptor to a first memory location, in a memory separate from the network interface adapter (Figure 2, system memory 38 or off-chip memory 67), indicating the data to be read from the system memory responsive to the incoming read request packet (page 19 line 27 through page 20 line 4, page 25 lines 6-10). The outgoing packet generator is adapted to read the response descriptor from the first memory location and, responsive thereto, to read the indicated data and to generate the outgoing response packet containing the indicated data (page 20 lines 12-15, page 26 lines 7-11). The outgoing packet generator comprises a gather engine (send data engine 66) that is coupled to gather both the write data and the read data from system memory for inclusion in the respective outgoing packets (page 20 lines 12-15, page 23 lines 3-7, page 26 lines 7-11). To submit the request, the host processor writes a request descriptor indicative of the write data to a second memory location (page 19 lines 12-14). The gather engine is adapted to read information from the response descriptor and from the request descriptor and to gather the read data and the write data responsive thereto (page 20 lines 13-15, page 23 lines 3-7, page 26 lines 7-11).

Dependent claim 65 adds to independent claim 1 the limitations that the incoming read request packet is a RDMA read request packet (page 19 lines 27-31) and that the response descriptor is a quasi-WQE (page 19 lines 27-31, page 21 lines 27-29, page 25 lines 6-11).

Independent claim 31 is directed toward a method of coupling a host processor (Figures 1 and 2, CPU 24) to a network (Figures 1 and 2, IB fabric 26). An outgoing packet generator (Figure 2, execution engine 60 and send data engine 66) is used to generate an outgoing request packet for delivery to a remote responder (Figure 1, remote host 30 or I/O device 34), responsive to a request submitted by the host processor (page 19 lines 19-24, page 20 lines 12-15, page 21 lines 8-10, page 22 line 30 through page 23 line 9). The outgoing request packet from the output packet generator is transmitted over the network to the remote responder (page 20 lines 15-19, page 21 line 29 through page 22 line 1). An incoming packet processor (Figure 2, transport check unit 52 and receive data engine 56) is used to receive an incoming response packet from the remote responder in response to the outgoing request packet sent thereto (page 18 line 29 through page 19 line 7). The incoming packet processor also is used to receive an incoming request packet sent by a remote requester (Figure 1, remote host 30) (page 18 line 29 through page 19 line 7). The incoming packet processor is coupled to the outgoing packet generator so as to cause the outgoing packet generator to generate, responsive to the incoming request packet, in addition to the outgoing request packet, an outgoing response packet for transmission via the network to the remote requester (page 19 line 27 through page 20 line 7). Generating the outgoing request packet comprises generating an outgoing write request packet containing write data taken from a system memory (Figures 1 and 2, system memory 38) associated with the host processor (page 19 lines 12-19, page 23 lines 3-7).

Coupling the incoming packet processor to the outgoing packet generator comprises using the outgoing packet generator to generate an outgoing read response packet containing read data taken from the system memory in response to the incoming request packet (page 20 lines 12-15, page 21 lines 15-16). Receiving the incoming request packet comprises receiving an incoming read request packet specifying data to be read from a system memory associated with the host processor (page 3 lines 1-3, page 25 lines 6-10). The coupling of the incoming packet processor also comprises writing, in response to the read request packet, a response descriptor to a first memory location of the system memory, indicating the data to be read therefrom (page 19 line 27 through page 20 line 4, page 25 lines 6-10) and causing the outgoing packet generator to read the response descriptor from the first memory location and, responsive thereto, to read the indicated data from the system memory and to generate the outgoing response packet containing the indicated data (page 20 lines 12-15, page 26 lines 7-11). The generating of the outgoing write request packet and the generating of the outgoing read response packet comprise using a gather engine (send data engine 66), in the outgoing packet generator, which is coupled to gather both the write data and the read data from the system memory for inclusion in the respective outgoing packets, to generate the packets (page 20 lines 13-15, page 23 lines 3-7, page 26 lines 7-11). Generating the outgoing write request packet comprises generating a request descriptor indicative of the write data in a second memory location (page 19 lines 12-14). Using the gather engine to generate the packets comprises reading information from the response descriptor and from the request descriptor and gathering the read data and the write data responsive thereto (page 22 line 30 through page 23 line 7, page 26 lines 7-11).

10

Dependent claim 66 adds to independent claim 31 the limitations that the incoming read request packet is a RDMA read request packet (page 19 lines 27-31) and that the response descriptor is a quasi-WQE (page 19 lines 27-31, page 21 lines 27-29, page 25 lines 6-11).

VI.   GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether claims 1, 4-9, 11, 12, 14, 16-19, 31, 34-39, 41, 42, 44, 46-49 and 64-66 are unpatentable over Pettey et al., US Patent No. 6,594,712 (henceforth, "Pettey") in view of Gasbarro, US Patent No. 6,948,004 (henceforth, "Gasbarro"), contrary to § 103(a).

VII.   ARGUMENTS

VIIA ARGUMENTS - REJECTIONS UNDER 35 U.S.C. 103a

The Examiner has rejected claims 1, 4-9, 11, 12, 14, 16-19, 31, 34-39, 41, 42, 44, 46-49 and 64-66 under § 103(a) as being unpatentable over Pettey in view of Gasbarro.

Pettey teaches an InfiniBand target channel adapter (TCA) **202** of an InfiniBand I/O unit **108** that communicates with a host **102** via an InfiniBand fabric **114**.

Gasbarro teaches a host fabric adapter **120** that a host system **130** uses for communicating with other systems via an InfiniBand switched fabric network **100'**.

Both Pettey's host system **102** and Gasbarro's host system **130** instruct their respective fabric adapters (TCA **202** or host fabric adapter **120**) to send packets to their respective InfiniBand networks (InfiniBand fabric **114** or switched fabric network **100'**) as described in the Background section of the above-identified patent application on page 2 lines 8-15:

> To send and receive communications over the network, the client initiates work requests (WRs), which causes work items, called work queue elements (WQEs), to be placed onto the appropriate queues. The channel adapter then executes the work items, so as to communicate with the corresponding QP of the channel adapter at the other end of the link.

In the case of Pettey (column 11 lines 21-24),

> When the CPU **208** of FIG. 2 desires to send the host **102** a message, it submits a work request **722** to the TCA **202** Send Queue **714**. The TCA **202** creates a Work Queue Entry (WQE) and places the WQE on the Send Queue **714**.

In the case of Gasbarro (column 7 lines 33-38),

> Work requests submitted by a consumer in a form Work Queue Elements "WQEs" are posted onto appropriate work queues (WQs) from the host system **130** to describe data movement operations and

13

location of data to be moved for processing and/or transportation, via the switched fabric **100'**.

See also Gasbarro column 17 lines 49-52:

"WQEs" are posted onto appropriate work queues (WQs) by the host software of the host system **130** to describe data transfer operations, via the switched fabric **100'**.

These WQEs correspond to the "request descriptors" recited in claims 1 and 33.

In the third paragraph on page 8 of the Office Action mailed September 25, 2008, and again in the fifth paragraph on page 12 of the Office Action mailed May 28, 2009, the Examiner cited column 11 line 18 through column 12 line 45 of Pettey et al. '712 as teaching the limitation, recited in independent claims 1 and 31, of writing a response descriptor to a first memory location indicating the data to be read responsive to an incoming read request packet. On the first line of page 3 of the Office Action mailed September 25, 2008, the Examiner more specifically identified the response descriptor recited in claims 1 and 31 with Pettey's scatter/gather list (SGL).

In the fifth paragraph on page 8 of the Office Action mailed September 25, 2008, and again in the first full paragraph on page 13 of the Office Action mailed May 28, 2009, the Examiner cited Gasbarro as teaching the limitation, recited in independent claims 1 and 31, of having a gather engine read information from the response descriptor and gather the read data responsive thereto. More specifically, in this context, the Examiner cited Gasbarro column 8 lines 28-41 and column 12 line 32 through column 13 line 36 as teaching the use of WQEs as descriptors for indicating data to be gathered. The Examiner has proposed that it would be obvious to combine the teachings of Pettey and Gasbarro to obtain the network interface adapter recited in claim 1 and the method recited in claim 31.

14

Accepting only for the sake of argument the Examiner's identification of Pettey's SGL with the response descriptor recited in claims 1 and 31, the Examiner has failed to make a *prima facie* case for the obviousness of claims 1 and 31 because the proposed combination of the teachings of Pettey and Gasbarro would be inoperative. Essentially, the Examiner proposed using Pettey's SGL as a Gasbarro WQE. But Pettey's SGL **900**, as illustrated in Pettey Figure 9 and as described in Pettey column 12 lines 24-36, is only "a portion of the WQE **800**" (Pettey column 12 line 26, emphasis added) and, as such, lacks essential information, such as destination QP **804** of WQE **800** as illustrated in Pettey Figure 8, that would be needed by Gasbarro if Pettey's SGL **900** were to be used by Gasbarro in the manner proposed by the Examiner. As stated in Gasbarro column 7 lines 37-39 and column 17 lines 52-54 with regard to WQEs,

> Such "WQEs" typically provide all the information needed to complete
> Send Queue and Receive Queue operations. (emphasis added)

Arguments similar to these were presented in response to the Office Action mailed September 25, 2008. In reply, on pages 3-6 of the Office Action mailed May 28, 2009, the Examiner has cited a statement, made by Applicant in response to the Office Action mailed January 17, 2008, to the effect that neither Pettey nor Gasbarro teach, hint or suggest anything resembling the response descriptors recited in claims 1 and 31. The Examiner finds a contradiction between Applicant's denial, in response to the Office Action mailed January 17, 2008, of anything in Pettey or Gasbarro resembling the response descriptors recited in claims 1 and 31 and Applicant's argument, in response to the Office Action mailed September 25, 2008, that Pettey's SGLs would be inoperative as Gasbarro's WQEs. Whether such a contradiction actually exists is moot because of Applicant's new demonstration that the Examiner's proposed combination of the teachings of Pettey and Gasbarro would be inoperative.

15

In addition, although Applicant now finds convincing the Examiner's argument, on page 8, first paragraph of the Office Action mailed September 25, 2008 and on page 12, third paragraph of the Office Action mailed May 28, 2009, that Pettey does teach, in column 14, the use of a response descriptor (the SGL) for responding to an incoming read request packet, Applicant continues to deny that Gasbarro teaches, hints or suggests anything resembling the use of a response descriptor for responding to an incoming read request packet. Indeed, Gasbarro teaches explicitly, in column 7 lines 47-50, *against* the use of a WQE in responding to RDMA read request packets:

> For an RDMA operation, the WQE also specifies the address in the remote consumer's memory. Thus <u>an RDMA operation does not need to involve the receive work queue of the destination</u>. (emphasis added)

Note that the "WQE" in this citation is the WQE of the requestor, not the WQE of the responder. Hence, in addition to the inoperability of the Examiner's proposed combination of the teachings of Pettey and Gasbarro, independent claims 1 and 31 are additionally allowable because Gasbarro teaches against such a combination.

With independent claims 1 and 31 allowable in their present form it follows that claims 4-9, 11, 12, 14, 16-19, 34-39, 41, 42, 44, 46-49 and 64-66 that depend therefrom also are allowable.

Although claims 65 and 66 are allowable merely by virtue of depending from claims 1 and 31, Applicant respectfully points out another reason why these claims are allowable. These claims limit the incoming read request packets of claims 1 and 31 to RDMA read request packets and limit the response descriptors of claims 1 and 31 to quasi-WQEs. In rejecting claims 65 and 66, the Examiner cited Pettey column 11 lines 1-53 as teaching the recited limitations. Pettey column 11 lines 1-53 teaches no such thing. Pettey column 11 lines 1-53 describes TCA **202** acting as a *requestor*, not as a responder. For example, column 11 lines 21-24 state:

When the CPU **208** of FIG. **2** desires to send the host **102** a message, it submits a work request **722** to the TCA **202** Send Queue **714**. The TCA **202** creates a Work Queue Entry (WQE) and places the WQE on the Send Queue **714**. (emphasis added)

In particular (column 11 lines 31),

...RDMA Read WQE **763**...specify, among other things, a virtual address in host **102** memory **124** for data transfers with the I/O unit **108**.

I/O unit **108** is the requestor. Host **102** is the responder. RDMA Read WQE **763** is posted in Send Queue **714** of TCA **202** acting as a *requestor*, not as a responder. Even Receive WQEs **782** are *requestor* WQEs, not responder WQEs. As stated in column 11 lines 39-41,

Receive WQEs **782** are placed on the Receive Queue **716** when the CPU **208** submits a work request **724** to the TCA **202**. (emphasis added)

The Examiner also cited Gasbarro column 7 lines 33-67 as teaching the limitations recited in claims 65 and 66. Gasbarro column 7 lines 33-67 teaches no such thing. Gasbarro column 7 lines 33-67 describes host system **130** acting as a *requestor*, not as a responder. For example, column 7 lines 33-38 state:

Work requests submitted by a consumer in a form Work Queue Elements "WQEs" are posted onto appropriate work queues (WQs) from the host system **130** to describe data movement operations and location of data to be moved for processing and/or transportation, via the switched fabric **100'**. (emphasis added)

In particular, with regard to RDMA operations, column 7 lines 47-50,

For an RDMA operation, the WQE also specifies the address in the remote consumer's memory. Thus an RDMA operation does not need to involve the receive work queue of the destination.

state explicitly that only the *requestor* of an RDMA operation, and not the responder of an RDMA operation, posts a WQE. As noted above, the "WQE" in this citation is the WQE of the requestor, not the WQE of the responder.

17

The Examiner also cited Gasbarro column 13 line 15 to column 14 line 28 as teaching the limitations recited in claims 65 and 66. Gasbarro column 13 line 15 to column 14 line 28 teaches no such thing. The Examiner should have begun his citation at column 13 line 6, not column 13 line 15. Column 13 lines 6-14 describes the Send Queue of a Work Queue pair in general terms as an "initiator" (column 13 line 7). Then column 13 lines 15-19 describes the Receive Queue of a Work Queue pair in general terms as a "responder" (column 13 line 16). Only in column 13 lines 30-35 does Gasbarro say anything about posting WQEs to work queue pairs. As noted above, Gasbarro teaches, in column 7 lines 33-38 and in column 17 lines 49-52, that WQEs are posted by host system **130**, and not by an incoming packet processor as recited in claims 1 and 31 from which claims 65 and 66 depend. In particular, as noted above, Gasbarro teaches in column 7 lines 47-50 that only the *requestor* of a RDMA operation such as the RDMA read operation recited in claims 65 and 66, and not the responder of a RDMA operation, posts a WQE.

In response to these arguments, the Examiner cited, on page 8 of the Office Action mailed May 28, 2009, Gasbarro column 13 lines 16-20 as teaching the use of the receive queue of a work queue pair for responding to incoming RDMA read requests, and Gasbarro column 13 lines 31-37 as teaching that WQEs are posted to work queue pairs for, *inter alia*, RDMA read operations. But, as noted above, Gasbarro teaches against using WQEs to respond to incoming RDMA read requests. Therefore, the reference in Gasbarro to the use of WQEs in connection with RDMA reads can only be to the use of WQEs for *initiating* RDMA read requests, not in *responding* to RDMA read requests.

The Examiner then cited, on page 9 of the Office Action mailed May 28, 2009, Gasbarro lines 47-48, "For an RDMA operation, the WQE also specifies the

18

address in the remote consumer's memory." As noted above, the WQE in this citation

is the WQE of the *requester*, not a WQE of the responder.

Respectfully submitted,

_____

Mark M. Friedman
Attorney for Applicant
Registration No. 33,883

Date: December 7, 2009

VIII. APPENDIX OF CLAIMS INVOLVED IN THE APPEAL

The text of the claims on appeal is:

1.   A network interface adapter, comprising:

a host interface, for coupling to a host processor;

an outgoing packet generator, adapted to generate an outgoing request packet for delivery to a remote responder responsive to a request submitted by the host processor via the host interface;

a network output port, coupled to receive the request packet from the outgoing packet generator, so as to transmit the outgoing request packet over a network to the remote responder;

a network input port, for coupling to the network so as to receive an incoming response packet from the remote responder, in response to the outgoing request packet sent thereto, and further to receive an incoming request packet sent by a remote requester; and

an incoming packet processor, coupled to the network input port so as to receive and process both the incoming response packet and the incoming request packet, and further coupled to cause the outgoing packet generator, responsive to the incoming request packet, to generate, in addition to the outgoing request packet, an outgoing response packet for transmission via the network output port to the remote requester;

wherein the outgoing request packet comprises an outgoing write request packet containing write data taken from a system memory accessible via the host interface;

wherein the outgoing response packet comprises an outgoing read response packet containing read data taken from the system memory in response to the incoming request packet;

wherein the incoming request packet comprises an incoming read request packet specifying data to be read from a system memory accessible via the host interface;

wherein the incoming packet processor is adapted to write a response descriptor to a first memory location, in a memory separate from the network interface adapter, indicating the data to be read from the system memory responsive to the incoming read request packet;

wherein the outgoing packet generator is adapted to read the response descriptor from the first memory location and, responsive thereto, to read the indicated data and to generate the outgoing response packet containing the indicated data;

wherein the outgoing packet generator comprises a gather engine, which is coupled to gather both the write data and the read data from the system memory for inclusion in the respective outgoing packets; and

wherein to submit the request, the host processor writes a request descriptor indicative of the write data to a second memory location, and wherein the gather engine is adapted to read information from the response descriptor and from the request descriptor and to gather the read data and the write data responsive thereto.

4.   An adapter according to claim 1, wherein the outgoing packet generator comprises a plurality of schedule queues, and is adapted to generate the

outgoing request packet and the outgoing response packet responsive to respective entries placed in the schedule queues of the plurality of schedule queues.

5.   An adapter according to claim 4, wherein the network input and output ports are adapted to receive and send the incoming and outgoing packets, respectively, over a plurality of transport service instances, and

wherein the outgoing request packet and the outgoing response packet are associated with respective transport service instances among the plurality of transport service instances, and

wherein the outgoing packet generator is adapted to assign the transport service instances of the plurality of transport service instances to the schedule queues of the plurality of schedule queues based on service parameters of the transport service instances of the plurality of transport service instances, and to place the entries in the schedule queues of the plurality of schedule queues corresponding to the transport service instances, of the plurality of transport service instances, with which the incoming and outgoing packets are associated.

6.   An adapter according to claim 5, wherein the outgoing packet generator comprises:

one or more execution engines, which are adapted to generate the outgoing request packet and the outgoing response packet responsive to a list of work items respectively associated with each of the transport service instances of the plurality of transport service instances; and

a scheduler, which is coupled to select the entries from the plurality of schedule queues and to assign the transport service instances of the plurality of

transport service instances to the one or more execution engines for execution of the work items responsive to the service parameters.

7.   An adapter according to claim 5, wherein the transport service instances of the plurality of transport service instances comprise queue pairs.

8.   An adapter according to claim 4, wherein the outgoing packet generator comprises one or more doorbell registers, to which the host processor and the incoming packet processor write in order to place the entries in the schedule queues of the plurality of schedule queues.

9.   An adapter according to claim 4, wherein the incoming request packet comprises an incoming write request packet carried over the network on a reliable transport service, and wherein responsive to the incoming write request packet, the incoming packet processor is adapted to add an entry to the entries placed in the schedule queues of the plurality of schedule queues, such that responsive to the entry, the outgoing packet generator generates an acknowledgment packet.

11.   An adapter according to claim 1, wherein the incoming packet processor is configured so that when it receives an incoming write request packet containing write data to be written to a system memory accessible via the host interface after receiving the incoming read request packet, it conveys the write data to the host interface without waiting for execution of the response descriptor.

12. An adapter according to claim 1, wherein the incoming packet processor is configured so that when it receives an incoming write request packet containing write data to be written to a system memory accessible via the host interface before receiving the incoming read request packet, it prevents execution of the response descriptor until the write data have been written to the system memory.

14. An adapter according to claim 1, wherein the outgoing packet generator is adapted, upon generating the outgoing request packet, to notify the incoming packet processor to await the incoming response packet so as to write a completion message to the host interface when the awaited incoming response packet is received.

16. An adapter according to claim 1, wherein the incoming read request packet is one of a plurality of incoming read request packets, and wherein the incoming packet processor is adapted to write a list of corresponding response descriptors to the first memory location, each said response descriptor indicating the data to be read from the system memory responsive to the corresponding incoming read request packet, responsive to which the outgoing packet generator is adapted to generate a sequence of corresponding outgoing response packets.

17. An adapter according to claim 16, wherein the network input and output ports are adapted to receive and send the incoming and outgoing packets, respectively, over a plurality of transport service instances, and wherein the incoming packet processor is adapted to prepare the list of the response descriptors for each of the transport service instances of the plurality of transport service instances as a part

of a response database held for the plurality of the transport service instances in common.

18. An adapter according to claim 17, wherein the transport service instances of the plurality of transport service instances comprise queue pairs.

19. An adapter according to claim 1, wherein the request comprises a write request, which is submitted by the host processor by generating a request descriptor indicating further data to be read from the system memory for inclusion in the outgoing request packet, and wherein the output packet generator is adapted to read the request descriptor and, responsive thereto, to generate the outgoing request packet as a write request packet containing the indicated further data.

31. A method for coupling a host processor to a network, comprising:

generating an outgoing request packet for delivery to a remote responder using an outgoing packet generator, responsive to a request submitted by the host processor;

transmitting the outgoing request packet from the output packet generator over the network to the remote responder;

receiving an incoming response packet from the remote responder, in response to the outgoing request packet sent thereto, using an incoming packet processor;

receiving an incoming request packet sent by a remote requester using the incoming packet processor; and

coupling the incoming packet processor to the outgoing packet generator so as to cause the outgoing packet generator to generate, responsive to the incoming request

packet, in addition to the outgoing request packet, an outgoing response packet for transmission via the network to the remote requester;

wherein generating the outgoing request packet comprises generating an outgoing write request packet containing write data taken from a system memory associated with the host processor;

wherein coupling the incoming packet processor to the outgoing packet generator comprises generating, using the outgoing packet generator, an outgoing read response packet containing read data taken from the system memory in response to the incoming request packet;

wherein receiving the incoming request packet comprises receiving an incoming read request packet specifying data to be read from a system memory associated with the host processor,

wherein coupling the incoming packet processor comprises writing, in response to the read request packet, a response descriptor to a first memory location of the system memory indicating the data to be read therefrom, and causing the outgoing packet generator to read the response descriptor from the first memory location and, responsive thereto, to read the indicated data from the system memory and to generate the outgoing response packet containing the indicated data;

wherein generating the outgoing write request packet and generating the outgoing read response packet comprise generating the packets using a gather engine in the outgoing packet generator, which is coupled to gather both the write data and the read data from the system memory for inclusion in the respective outgoing packets; and

wherein generating the outgoing write request packet comprises generating a request descriptor indicative of the write data to a second memory location, and

wherein generating the packets using the gather engine comprises reading information from the response descriptor and from the request descriptor using the gather engine and gathering the read data and the write data responsive thereto.

34. A method according to claim 31, wherein the outgoing packet generator comprises a plurality of schedule queues, and wherein generating the packets comprises generating the outgoing request packet and the outgoing response packet responsive to respective entries placed in the schedule queues of the plurality of schedule queues.

35. A method according to claim 34, wherein the outgoing request packet and the outgoing response packet are associated with respective transport service instances among a plurality of transport service instances in use on the network, and wherein generating the outgoing request packet and the outgoing response packet comprises assigning the transport service instances of the plurality of transport service instances to the schedule queues of the plurality of schedule queues based on respective service parameters of the transport service instances of the plurality of transport service instances, and placing the entries in the schedule queues, of the plurality of schedule queues, corresponding to the transport service instances, of the plurality of transport service instances, with which the packets are associated.

36. A method according to claim 35, wherein generating the outgoing request packet and the outgoing response packet comprises allocating resources to process the schedule queues, of the plurality of schedule queues, responsive to the respective service parameters.

37.   A method according to claim 35, wherein the transport service instances of the plurality of transport service instances comprise queue pairs.

38.   A method according to claim 34, wherein receiving the incoming request packet further comprises receiving an incoming write request packet on a reliable transport service, and wherein generating the outgoing response packet comprises adding an entry to the entries in the schedule queues of the plurality of schedule queues, causing the outgoing packet generator, responsive to the entry, to generate an acknowledgment packet.

39.   A method according to claim 34, wherein generating the outgoing write request packet and generating the outgoing read response packet both comprise writing to doorbell registers of the outgoing packet generator in order to place the entries in the schedule queues of the plurality of schedule queues.

41.   A method according to claim 31, wherein receiving the incoming request packet further comprises receiving an incoming write request packet containing write data to be written to a system memory associated with the host processor after receiving the incoming read request packet, and comprising conveying the write data to the system memory using the incoming packet processor without waiting for execution of the response descriptor associated with the outgoing read response packet.

42.   A method according to claim 31, wherein receiving the incoming request packet further comprises receiving an incoming write request packet

containing write data to be written to a system memory associated with the host processor before receiving the incoming read request packet, and comprising conveying the write data to the system memory using the incoming packet processor while preventing execution of the response descriptor associated with the outgoing read response packet until the write data have been written to the system memory.

44.   A method according to claim 31, wherein transmitting the outgoing request packet comprises passing a notification from the output packet generator to the incoming packet processor to await the incoming response packet to be received in response to the outgoing request packet, and comprising writing a completion message to the host processor when the incoming packet processor receives the awaited packet.

46.   A method according to claim 31, wherein receiving the incoming read request packet comprises receiving a plurality of incoming read request packets, and wherein writing the response descriptor comprises writing a list of said response descriptors to the first memory location, causing the outgoing packet generator to generate a sequence of corresponding outgoing response packets containing respective indicated data.

47.   A method according to claim 46, wherein receiving the plurality of incoming read request packets comprises receiving the plurality of incoming read request packets over a plurality of transport service instances on the network, and wherein writing the list of the descriptors comprises writing a respective list for each transport service instance of the plurality of the transport service instances to a

response database held for the plurality of the transport service instances in common, causing the outgoing packet generator to generate the packets for transmission over the plurality of the transport service instances.

48.   A method according to claim 47, wherein the transport service instances of the plurality of transport service instances comprise queue pairs.

49.   A method according to claim 31, wherein the request comprises a write request, which is submitted by the host processor by generating a request descriptor in a second memory location indicating further data to be read from the system memory for inclusion in the outgoing request packet, and wherein generating the outgoing request packet comprises reading the request descriptor from the second memory location and, responsive thereto, generating a write request packet containing the indicated further data.

64.   An adapter according to claim 1, wherein the memory separate from the network interface adapter is the system memory.

65.   An adapter according to claim 1, wherein the incoming read request packet is a RDMA read request packet and wherein the response descriptor is a quasi-WQE.

66.   A method according to claim 31, wherein the incoming read request packet is a RDMA read request packet and wherein the response descriptor is a quasi-WQE.

IX. APPENDIX OF EVIDENCE

NONE

X.   APPENDIX OF RELATED PROCEEDINGS

NONE